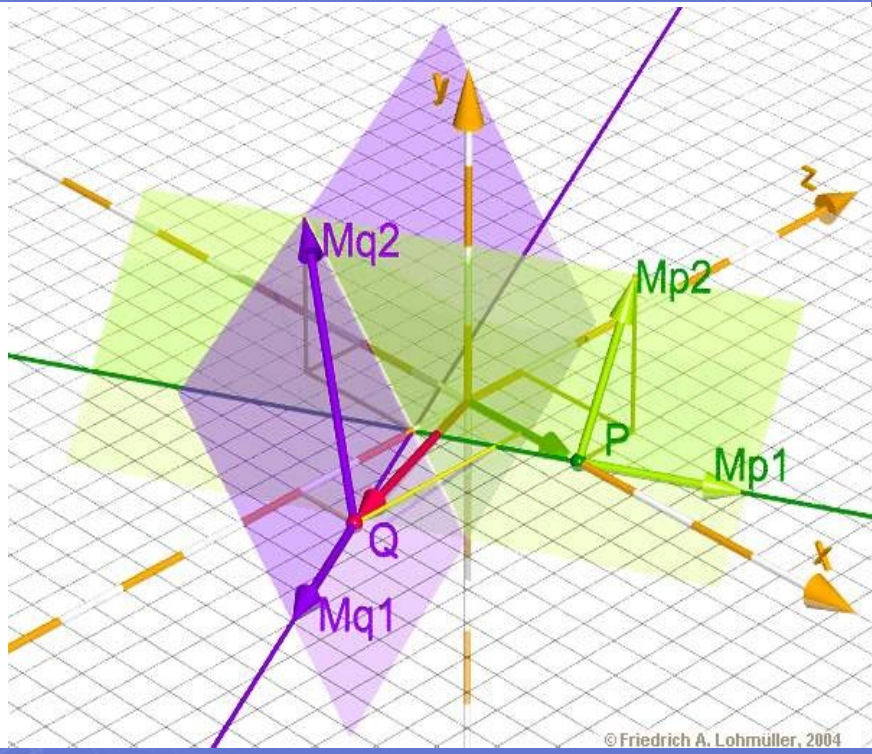
Raytracing on the Cell Broadband Engine

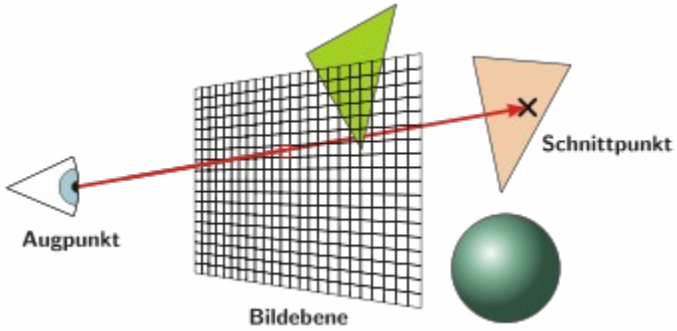
A quick overview



Agenda

- Introduction
- Objectives of the work
- Cell Processor
- Ray Tracing Application Architecture
- Summary

Introduction



- **What is Raytracing?**
  - Method to create photo-realistic images on a computer
  - Rays are casted through an imaginary camera into a virtual world
  - Raytracing mathematically models the light rays as they bounce in the virtual world
  - Advanced optical effects
  - Accurate simulations of reflection and refraction
  - Rendering time differs from seconds to many days
  - The waiting is worth while -> Impressive results

Introduction

- **Why Raytracing?**
  - Vector arithmetic
  - Nearly only floating-point calculation
  - Highly parallelizable
  - Every one of the rays is independent from each other
  - It's up to the developer how much work to distribute
  - The smallest amount of work can be a
    - Single ray
    - Raypacket (2x2, 4x4 ...)
    - Tile ...


Christoph Hormann <chris_hormann@gmx.de>


[INCUBUS | FINAL]

Objectives of the work

- **Solution to an increasing demand of computational power**
  - Manufactures take multi-core chip design into consideration
- **Consequences**
  - A paradigm shift is to take place
  - To solve a specific task entirely new approaches have to be developed
  - The tasks have to be adapted to the new architecture
  - A variety of approaches have to be taken into account to solve a given problem
  - With respect to its performance and adaptability each approach has to be tested to determine the best solution

Objectives of the work

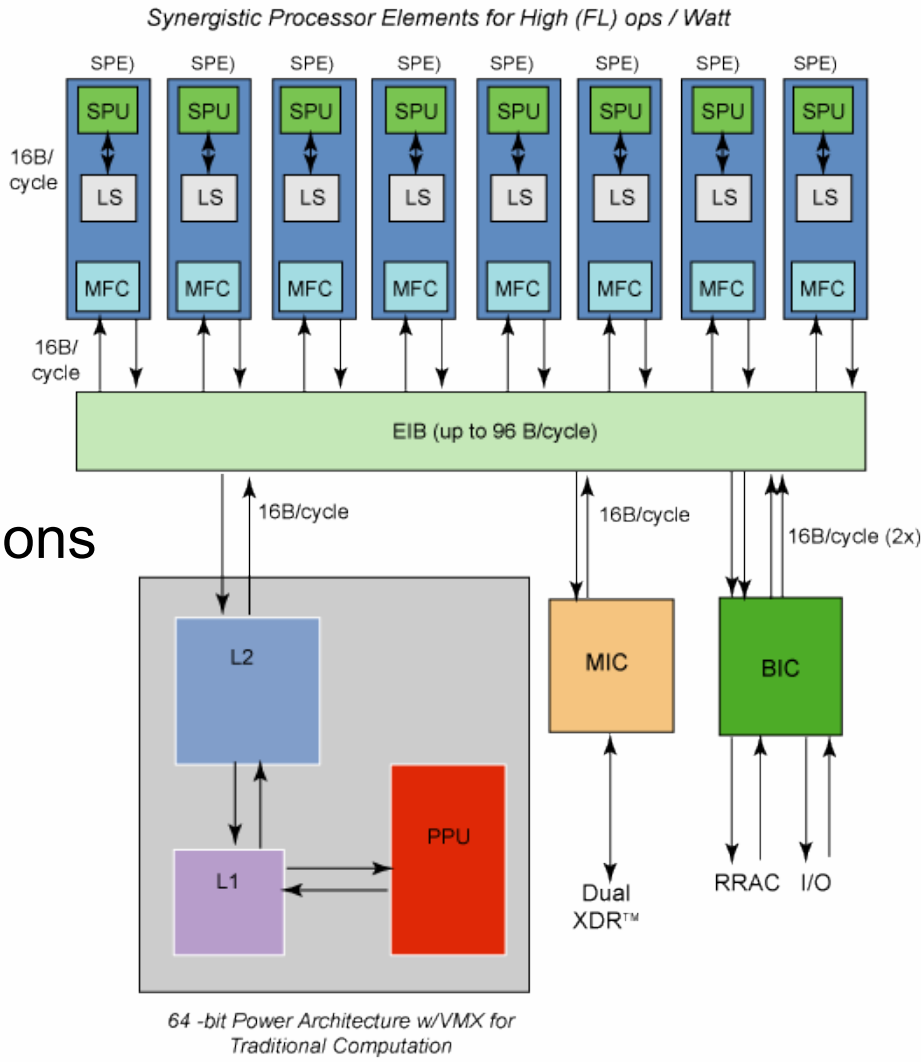
- **Diploma thesis**
  - The problem that yields out of that situation will be examined for a Raytracer which will be adapted to the CBE architecture
  - Different approaches are supposed to be introduced
  - The single approaches are examined with its respect to performance and its capability for Raytracing
  - Goal of the work is to develop an optimal architecture for Raytracing for the CBE with focus on performance
  - Result of the work will be a Raytracer that can render medium sized scenes at acceptable rates

Cell Processor

- **Joint Venture** of Sony Toshiba and IBM also known as **STI**, formally started 2001
- Cell Processor first incarnation of a new family of microprocessors conforming the **Cell Broadband Engine Architecture**
- Initially attended for applicatons in game consoles and media-rich consumer-electronic devices
  - Sony Playstation 3, HDTV, Developer Boards, Cell Blades
- A much broader use of the architecutre envisioned
  - Cell designed to enable fundamental advances in processor performance

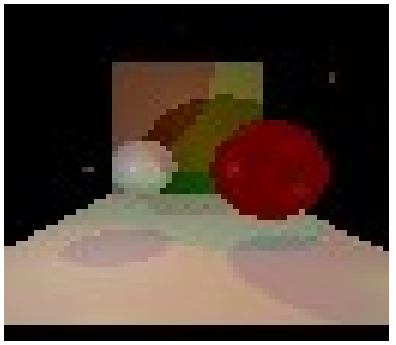
Cell Processor Highlights

- **9 Multi-Core microprocessor**
  - 1 PowerPC Processing Element
  - 8 Synergetic Processing Elements
- **3.2 Ghz** clock frequency
- **10x** performance for many applications



Previous Work

- **Raytracer** based upon the recursive ray tracing algorithm
  - Featured reflection but no refraction
  - Performance as a Pentium 4
- **Streaming programming model**
  - Data-Flow not optimal
- **Drawbacks**
  - Poor performance
  - Shader, Illumination modell not easily extensible
  - Poor use of the hardware architecture

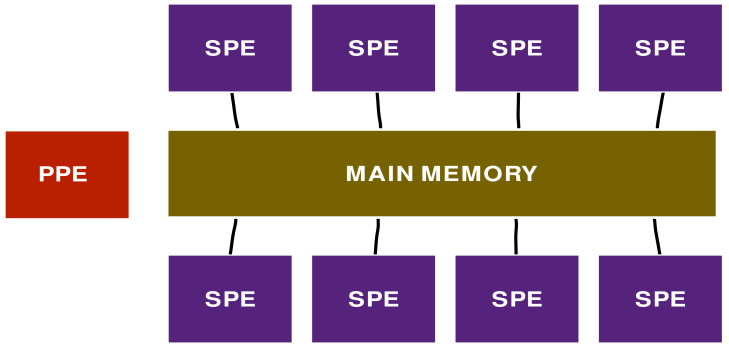


Targeting the CBE

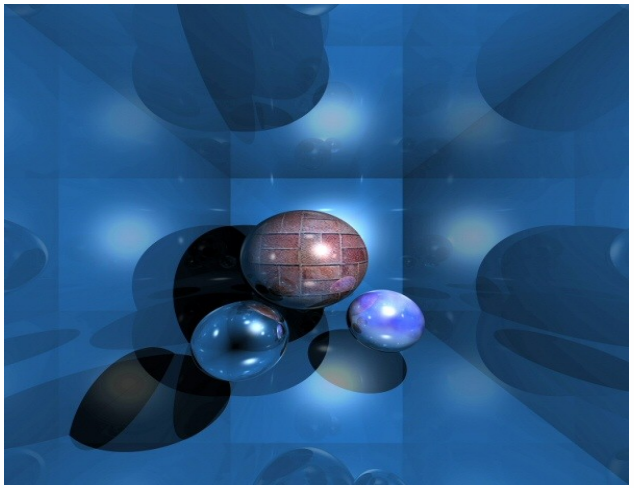
- **General work flow**
  - Porting to the PPE
  - Application partitioning, Programming model
  - Offloading parts to the SPE's
  - Using SPU C/C++ Language Extensions to achieve maximum performance
  - Tuning data access and flow with double-buffering and prefetching
  - SIMD vectorization

Ray Tracing Application Architecture (CBE view)

- **Shared memory programming model**
  - Access to main memory reduced
  - Communication overhead minimized
  - PPE needed just for initialization
  - 4 -Way Set Associative Cache
  - Load Balancing



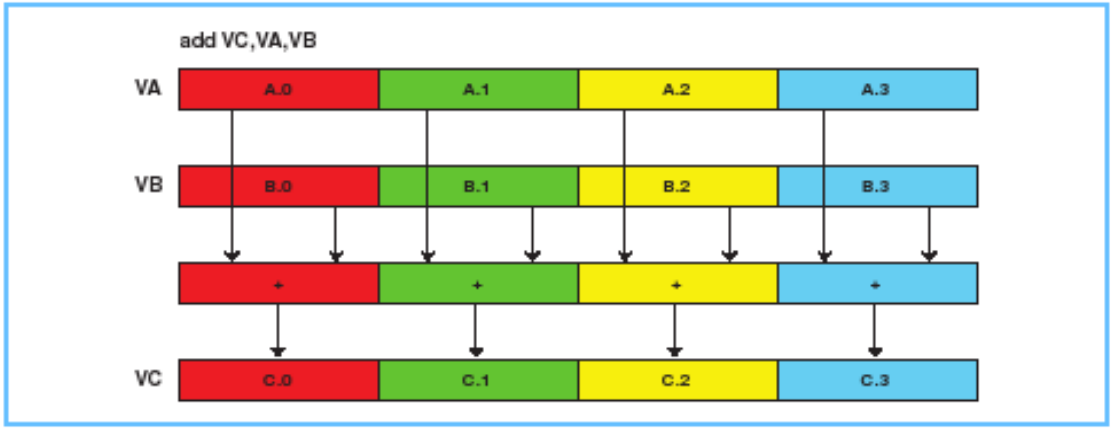
Ray Tracing Application Architecture (SPE view)



- **Every SPE a little Ray Tracer**
  - Raytracing acceleration techniques
  - Input camera, objects
  - Output rendered tile, access to main memory reduced
  - PPE completely ignored
  - Output is double-buffered into a SDL Surface
  - Communication between the SPE's not needed

Ray Tracing Application Architecture (SPU view)

- **Data Flow and SIMD vectorization**
  - Double buffering and prefetching
  - Loop vectorization
  - Ray packets, simultainously perform calculation on 4 rays
  - Caching of often used objects



Ray Tracing Application Architecture

- **Shared memory programming model**
  - Reducing the access to main memory
  - Shader pluggable
- **Cache management**
  - 4 – Way Set Associative Cache
- **Load Balancing**
  - Dimension Exchange method (Work Stealing)

Summary

- **Raytracing Application Architecture**
  - Outperformed the last implementation
  - Shader easily pluggable
  - Options for further improvements
- **CBE**
  - Fast , powerful
  - Unleashing the power, needs knowledge about the hardware